



PPC750FX Evaluation Board Kit

User's Manual

SA14-2730-05

September 04, 2003

PowerPC®





© Copyright International Business Machines Corporation 2003

All Rights Reserved

Printed in the United States of America September 2003

This version of the *750FX Evaluation Board Kit User's Manual* applies to Version 1.0 of the 750FX Evaluation Kit Software and all subsequent versions until otherwise indicated in new versions or technical newsletters.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both.

IBM

IBM Logo

CoreConnect

PowerPC

PowerPC logo

PowerPC Architecture

RISCTrace

RISCWatch

Other company, product, and service names may be trademarks or service marks of others.

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in implantation, life support or other hazardous uses where malfunction could result in death, bodily injury, or catastrophic property damage. The information contained in this document does not affect or change IBM product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

© Copyright 1982, 1986, 1988, 1990, 1994 Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the University of California, Berkeley and its contributors.
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.



PPC750FX Evaluation Board Kit

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY 12533-6351

The IBM home page can be found at <http://www.ibm.com>

The IBM Microelectronics Division home page can be found at <http://www.ibm.com/chips>

title.fm.
September 04, 2003

Contents

Figures	7
Tables	9
About This Book	11
Who Should Use This Book	11
How to Use This Book	12
Contacting the IBM Embedded Systems Solution Center	12
Related Publications	12
Part I. Evaluation Board Kit Installation Setup	15
1. Evaluation Board Kit Overview	17
Hardware Components	17
Evaluation Board Platform	17
Cables and Power Supply	17
Software Components	18
BSP Software	18
PowerPC Initialization Boot Software (PIBS)	18
Embedded PowerPC Operating System (EPOS)	18
Dhrystone Benchmark Program	18
Application Tools	18
RISCWatch Debugger	18
PowerPC Development Tools/Utilities	19
2. Host System Requirements	21
Host System Requirements	21
3. Installing the Software	23
What is installed	23
Software Installation	23
Software Installation	23
4. Host Configuration	25
Serial Port Setup	25
Ethernet Setup	25
5. Board Connectors	27
Connecting the Evaluation Board to the Host	27
Using a Terminal Emulator	29
Terminal Emulation	29
Board Reset	30
Part II. PowerPC Initialization Boot Software (PIBS)	31
6. PIBS Overview	33
PPC750FX External Hardware Support	33
Quick Start	33
PIBS Memory Map	34

7. PIBS Commands	35
8. PIBS Environment Variables	57
9. PIBS Software Development	70
Directory Structure	70
Build Tools	71
Makefile Setup	71
PIBS Build Process	71
Programming PIBS2 Software in FLASH	72
Dual Boot Capability	73
10. Configuring TFTP	74
11. IBM Image File Format Description	75
Boot Header	75
Section Headers	75
Part III. Sample Applications	79
12. Sample Applications Overview	81
13. How to Load Application Programs	83
14. EPOS Shell and Dhrystone Sample	85
15. MultiProcessor Sample	87
16. Debugging Application Programs	89
Index	91
Revision Log	93

Figures

Figure 5-1.	Serial Port Connection	27
Figure 5-2.	Signal Assignments in a Crossover Cable	28
Figure 5-3.	Point-to-Point Ethernet Connection	28
Figure 5-4.	Ethernet Connection with Hub	29
Figure 14-1.	Sample Output from EPOS	85



Tables

Table 6-1.	PIBS Memory Map	34
Table 7-1.	Board Configuration Information	38
Table 8-1.	PIBS System Initialization Environment Variables	57
Table 8-2.	PIBS Runtime Environment Variables	57
Table 9-1.	PIBS directory tree	70
Table 9-2.	Files produced from a make pibs2	72
Table 9-3.	Files produced by make util command	72
Table 9-4.	Files needed to install PIBS onto the PPC750FX platform using flash_pibs.cmd	73



About This Book

This book contains the information you need to install and use the IBM® PowerPC 750FX Evaluation Board Kit, a hardware and software development tool for the PowerPC PPC750FX 32-bit microprocessor.

The PowerPC 750FX Evaluation Board Kit hardware includes the PowerPC 750FX Evaluation Board, power supply, line cord, and board interface cables. Features of the evaluation board include dual PPC750FX processors, the Marvell MV64360 system controller, 256MB DDR SDRAM with ECC, 1MB 8-bit socketed flash, 32MB 32-bit FLASH, 1MB SRAM, 32KB non-volatile FRAM, two Ethernet ports, two serial ports, two 64kb IIC serial EEPROMs.

The PPC750FX evaluation board kit software includes the IBM PowerPC Initialization Boot Software (PIBS) resident in the flash memory on the board, PIBS source code, the IBM Embedded PowerPC Operating System (EPOS), sample application programs, application development libraries and tools, compiler/linker/assembler and associated binary utilities, and IBM's RISCWatch, a source-level debugger that runs on the host system.

The PPC750FX evaluation board kit also includes technical specifications and board schematics.

Connection of the evaluation board to a host system is required for the exercises in this book.

Who Should Use This Book

This book is for hardware and software developers who need to evaluate the PowerPC PPC750FX microprocessor and use the debugging features of the PowerPC 750FX Evaluation Board Kit to support software development.

Users should understand hardware and software development tools, concepts, and environments. Specifically, users should understand:

- The host's operating system
- The PowerPC Architecture™ and implementation-specific characteristics of the PowerPC microprocessor being used
- C and Assembler language programming

PPC750FX Evaluation Board Kit

How to Use This Book

This book contains the following sections organized in three parts:

- Part I Evaluation Board Kit Installation Setup
 - *Evaluation Board Kit Overview* on page 17
 - *Host System Requirements* on page 21
 - *Installing the Software* on page 23
 - *Host Configuration* on page 25
 - *Board Connectors* on page 27
- Part II PowerPC Initialization Boot Software (PIBS)
 - *PIBS Overview* on page 33
 - *PIBS Commands* on page 35
 - *PIBS Environment Variables* on page 57
 - *PIBS Software Development* on page 70
 - *Configuring TFTP* on page 74
 - *IBM Image File Format Description* on page 75
- Part III Sample Applications
 - *Sample Applications Overview* on page 81
 - *How to Load Application Programs* on page 83
 - *EPOS Shell and Dhrystone Sample* on page 85
 - *Debugging Application Programs* on page 89

To help readers find material in these chapters, the book contains:

- *Contents* on page 5
- *Figures* on page 7
- *Tables* on page 9
- *Index* on page 91

Contacting the IBM Embedded Systems Solution Center

For information about the PowerPC 750FX Evaluation Board Kit and the IBM family of hardware and software products for embedded system developers, check out the IBM Microelectronics web site at:

<http://www.chips.ibm.com/products/powerpc>

Please send any comments or questions regarding this product to the following Internet address:

ppcsupp@us.ibm.com

Related Publications

Many of the following publications are included on the CD ROM that comes with the evaluation kit or can be found on the IBM Microelectronics technical library website at:

<http://www.chips.ibm.com/techlib>

The others are available from your IBM Microelectronics representative:

- Embedded Application Binary Interface (EABI) Publications

PowerPC Embedded Application Binary Interface (EABI)

System V Application Binary Interface, Third Edition, 0-13-0100439-5

System V Application Binary Interface, PowerPC Processor Supplement

- RISCWatch Debugger Publications

RISCWatch Debugger User's Guide, 13H6964

RISCWatch Debugger Installation Guide, 13H6984

- PowerPC 750FX Embedded Processor Publications

PowerPC 750FX Embedded Processor Data Sheet

PowerPC 750FX Embedded Processor User's Manual

- Evaluation Board Publications

PowerPC 750FX Evaluation Board User's Manual

- EPOS Publications

Embedded PowerPC Operating System User's Manual





Part I. Evaluation Board Kit Installation Setup



1. Evaluation Board Kit Overview

This section introduces the hardware and software components included in the PPC750FX evaluation board kit.

1.1 Hardware Components

The PPC750FX evaluation board kit contains the evaluation board, power supply, line cord, 2 serial port cables and 1 crossover Ethernet cable.

1.1.1 Evaluation Board Platform

Features of the evaluation board include:

- PCI adapter form factor
- Two IBM PowerPC 750FX processors
- Marvell MV64360 System Controller
- 256MB DDR SDRAM with ECC
- 1MB 8-bit wide socketed FLASH (2 -512KB devices)
- 1MB 8-bit wide SRAM
- 32MB 32-bit wide FLASH
- 32KB Ferroelectric Nonvolatile RAM (FRAM)
- Two 100BASE-TX Ethernet ports
- Two 16550 compatible serial ports
- Two 64kb IIC Serial EEPROMs
- Single RISCWatch header for both processors
- Powered either externally or from PCI slot
- External system clock input
- External input for programming the on-board CPLD.

For a detailed description of the evaluation board, refer to the *PowerPC 750FX Evaluation Board Manual*.

1.1.2 Cables and Power Supply

The PPC750FX evaluation board kit includes 2 serial port cables for connecting the board's serial ports to the host system (See *Figure 5-1 Serial Port Connection* on page 27).

An Ethernet crossover cable is provided in the kit to support direct Ethernet communication with the host system. Two standard 10/100Mbps Ethernet connectors are provided on the evaluation board. The Ethernet crossover cable is for direct connection to a single host and *cannot* be used with a hub or a building's Ethernet network. The crossover cable supports 10Mb/s operation. For 100Mb/s operation a hub or switch (not supplied) should be used.

A power supply and line cord are also provided with the PPC750FX evaluation board kit to enable stand alone operation.

PPC750FX Evaluation Board Kit

1.2 Software Components

The PPC750FX evaluation board kit software consists of the Board Support Package (BSP), the RISCWatch source level debugger, and the PowerPC development tools.

1.2.1 BSP Software

The BSP software includes the PowerPC Initialization Boot Software (PIBS), the Embedded PowerPC Operating System (EPOS), sample programs (including the Dhrystone benchmark program), and tools. Source code is provided for PIBS and EPOS software

1.2.1.1 PowerPC Initialization Boot Software (PIBS)

The PIBS firmware is supplied in the 2 512KB socketed flash memory modules on the evaluation board. PIBS is accessed via a serial port connection between Serial Port 1 on the board (J13 Right) and a serial port on the host computer. PIBS initializes the PPC750FX processor and the board for serial and Ethernet communications. By supporting communications with the host computer system, PIBS allows applications to be loaded from the host onto the board.

See *PIBS Overview* on page 33 for additional information on PIBS.

1.2.1.2 Embedded PowerPC Operating System (EPOS)

EPOS is a real-time operating system developed by IBM for its PowerPC processors. EPOS can be configured to meet the functional requirements and memory constraints of a wide variety of embedded applications. PIBS and sample programs are EPOS applications. See the *Embedded PowerPC Operating System User's Manual* for additional information on EPOS.

1.2.1.3 Dhrystone Benchmark Program

The Dhrystone benchmark is a commonly available integer benchmark. It is included as a sample program to be built, loaded onto the board, and executed. The results of this benchmark may vary based on the compiler used, compiler options, and the system environment in which it is run.

1.2.1.4 Application Tools

Several host-based tools are provided to support application development for the evaluation board.

1.2.2 RISCWatch Debugger

The RISCWatch source level debugger provides a window-based debugging environment debugging application programs on the board. Debugger installation and usage are addressed in the *RISCWatch Debugger Installation Guide* and the *RISCWatch Debugger User's Guide*. These documents are included on the evaluation kit CD-ROM included with the kit.

The RISCWatch debugger runs on the host system, communicating with Debug Monitor running on the evaluation board through the Ethernet interface.

1.2.3 PowerPC Development Tools/Utilities

The PPC750FX evaluation board kit is shipped with one of the two compiler options.

The IBM HighC C++ Compiler contains a HighC/C++ compiler, assembler, linker and various other utilities for application development. The IBM HighC C++ Compiler can be activated for a 30 day evaluation period. After 30 days a compiler license must be purchased. Please consult the **readme.txt** file that is included with the compiler for details on compiler licensing.

The GNU for PowerPC development tools contain the GNU compiler, assembler, linker, and various other utilities for application development.

The IBM HighC C++ Compiler or the GNU for PowerPC development tools can be used to compile code for execution on the evaluation board.



2. Host System Requirements

This section describes the hardware and software requirements of the host system to which the evaluation board is to be connected. Supported host systems include:

- PC running one of the following:
 - Windows 98/NT/2000/ME

2.1 Host System Requirements

Recommended hardware requirements of the host PC include:

- Pentium class CPU 500MHz or faster with at least 64MB RAM
- SVGA, XGA Display Monitor (1024 x 768)
- Approximately 60MB of free disk space. This space is required for the PowerPC development tools, the Board Support Package software, and the RISCWatch debugger. When planning disk space usage, consider disk space requirements for Windows and any other software packages.
- At least one available serial port for terminal emulation. Establishing an Ethernet host-to-board connection may require the installation of an Ethernet adapter card on the host (if not already installed) and some additional connectivity hardware. That hardware may include any or all of the following:
 - An Ethernet 10/100 BaseT network transceiver, a twisted pair cable, and a hub. At a minimum, a point-to-point connection will require the Ethernet crossover cable supplied with the kit. The Ethernet crossover cable is for direct connection to a single host and *cannot* be used with a hub/switch or a building's Ethernet network. The crossover cable is only supported for 10Mb/s operation. For 100Mb/s a hub/switch (not supplied) should be used.



3. Installing the Software

This section describes the procedures for installing the evaluation board kit software on the host system.

3.1 What is installed

The evaluation board software consists of the Board Support Package (BSP), the RISCWatch debugger, and the PowerPC development tools. The BSP consists of the PowerPC Initialization Boot Software (PIBS) and the Embedded PowerPC Operating System (EPOS). Instructions for installing the RISCWatch Debugger software can be found in the *RISCWatch Debugger Installation Guide*.

Note: RISCWatch version 5.1 or greater is required for use with the PowerPC 750FX Evaluation Board.

3.2 Software Installation

3.2.1 Software Installation

Before beginning the installation, you must have:

- Evaluation Board Kit Software CD
- A PC running Windows 98/NT/2000/ME

The following procedure installs the evaluation kit software:

Note: For Windows NT users, we recommend that you log on as **administrator**.

1. Insert the CD labeled "PowerPC 750FX Embedded Processor Evaluation Kit Software" into the CD-ROM drive. A pop-up window should soon appear, allowing you to install each of the evaluation board kit software packages separately. If the window does not appear, follow the steps below.
2. Select **Start** from the Windows task bar.
3. Select **Run**.
4. Type **D:\kitinst.exe** then press **Enter** to run the installation program (where **D:** is your CD-ROM drive).
5. Follow the installation program instructions.

If the default install directory is accepted, the BSP software and PowerPC Development tools are installed in the **IBM** directory tree. The default install directory for RISCWatch is **Program Files\RISCWatch**.



4. Host Configuration

Several host configuration steps are required to facilitate communications between the host computer and the board. These steps are outlined in this chapter.

4.1 Serial Port Setup

Most PCs include two serial ports to support communications via asynchronous data transfer. These ports are sometimes referred to as communication or COM ports. These ports are usually accessed from the back of the system unit. You should consult your PC literature to determine how many serial ports are available on your unit and where they are located.

The connection of the terminal emulator running on the host to the PIBS monitor running on the board, is made through a serial port on the PC and Serial Port 1 (J13 Right) on the board. The host serial port must be configured for a baud rate of 115200, 8 data bits, 1 stop bit, and no parity. The proper setting of these parameters is discussed in the section on terminal emulation.

The serial port connection can be used to download applications from the host to the board but it can not be used to establish a debug connection with the RISCWatch debugger.

4.2 Ethernet Setup

An Ethernet connection can be used for host-to-board communications. The Ethernet connection is made through an Ethernet adapter on the host and a 10/100 BaseT Ethernet port (J20 Left) on the board. Ethernet is used when downloading applications on to the board or when using the RISCWatch debugger.

An Ethernet connection may require additional hardware. The evaluation board supports a standard Ethernet, twisted pair (10/100 BaseT) connection. This connection requires that the host PC be equipped with an appropriate Ethernet adapter. The host adapter is not included in the kit. Please consult your PC and adapter documentation for requirements and installation instructions.

At a minimum, a 10BaseT connection requires a crossover Ethernet twisted pair cable (included in the kit) for point-to-point communications. The Ethernet crossover cable is for 10Mb/s direct connection to a single host and *cannot* be used with a hub/switch or a building's Ethernet network. If you want more than two nodes, or 100Mbps connectivity, you will need a hub/switch and straight-through twisted pair cables.

Other hardware required will depend on the type of Ethernet adapter you have on your PC and whether the board is being connected to an existing Ethernet network. Please consult your system administrator and the documentation included with the adapter hardware for additional instructions.

Establishment of an Ethernet interface requires a host IP address. If the host PC is connected to an existing Ethernet network, the host IP address should already be defined and there is no need to set it again. Consult your network administrator on how to obtain the host's Ethernet IP address and how to add the board to the existing network.

To set the host IP address for the Ethernet connection:

1. Select the My Computer icon from the desktop.
2. Select Control Panel.
3. Select Network.

PPC750FX Evaluation Board Kit

4. Add the appropriate Adapter network component for the Ethernet adapter being used (if not already added).
5. Add a Protocol network component of Microsoft - TCP/IP' (if not already added). Specify the IP address and netmask to be used.

For the update to take effect, TCP/IP may need to be restarted. This may require a reboot of the system and/or a restart of TCP/IP. Make a note of the host IP address assigned to the Ethernet adapter, as this value will need to be made known to the PIBS software.

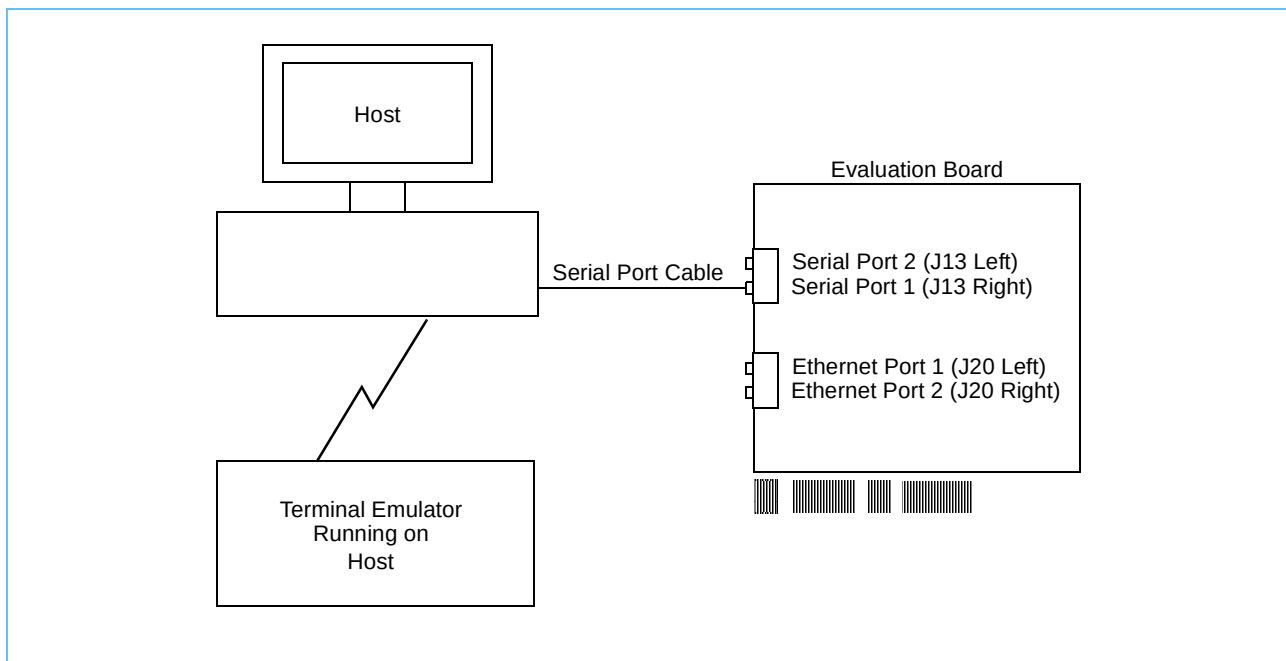
5. Board Connectors

For detailed descriptions of the connectors and jumpers on the evaluation board, please refer to the *PowerPC 750FX Evaluation Board User's Manual*.

5.1 Connecting the Evaluation Board to the Host

To establish a working environment, the evaluation board must be connected to a host system. PowerPC Initialization Boot Software (PIBS) access requires a connection between Serial Port 1 on the board (J13 Right) and a serial port on the host as shown in *Figure 5-1*. Users can download but not debug application code using the serial port connection. An Ethernet connection must be established when using the RISC-Watch debugger to debug an application running on the evaluation board.

Figure 5-1. Serial Port Connection



Assuming a terminal emulator running on the host is going to be used for PIBS access, use the cable provided with the kit to connect the Serial Port 1 connector (J13 Right) on the board to a serial port on the host. The host end might require a serial port adapter (not supplied) for connectivity. Serial port 2 can be connected to the host if the demo application for the CPU1 will be executed.

The Ethernet connection can be made in one of two ways. If the connection is to be used exclusively between the host and the board, and only 10Mb/s speed is required, the crossover cable provided with the kit can be used to directly connect the two nodes in a point-to-point Ethernet connection. Otherwise, a 10/100BaseT hub/switch (not provided) must be used to connect the nodes together.

Note: The Ethernet 10BaseT crossover cable supplied with the kit will **not** work if plugged into a hub. Ethernet straight-through cables (not included in the kit) are required when using a hub or a switch.

PPC750FX Evaluation Board Kit

Figure 5-2 shows the connections and signal assignments required in a crossover cable.

Figure 5-2. Signal Assignments in a Crossover Cable

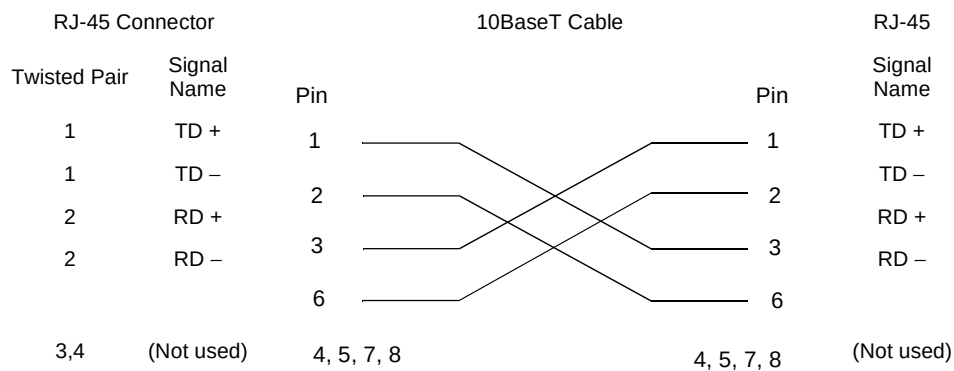


Figure 5-3 shows a point-to-point Ethernet connection using the provided crossover cable.

Figure 5-3. Point-to-Point Ethernet Connection

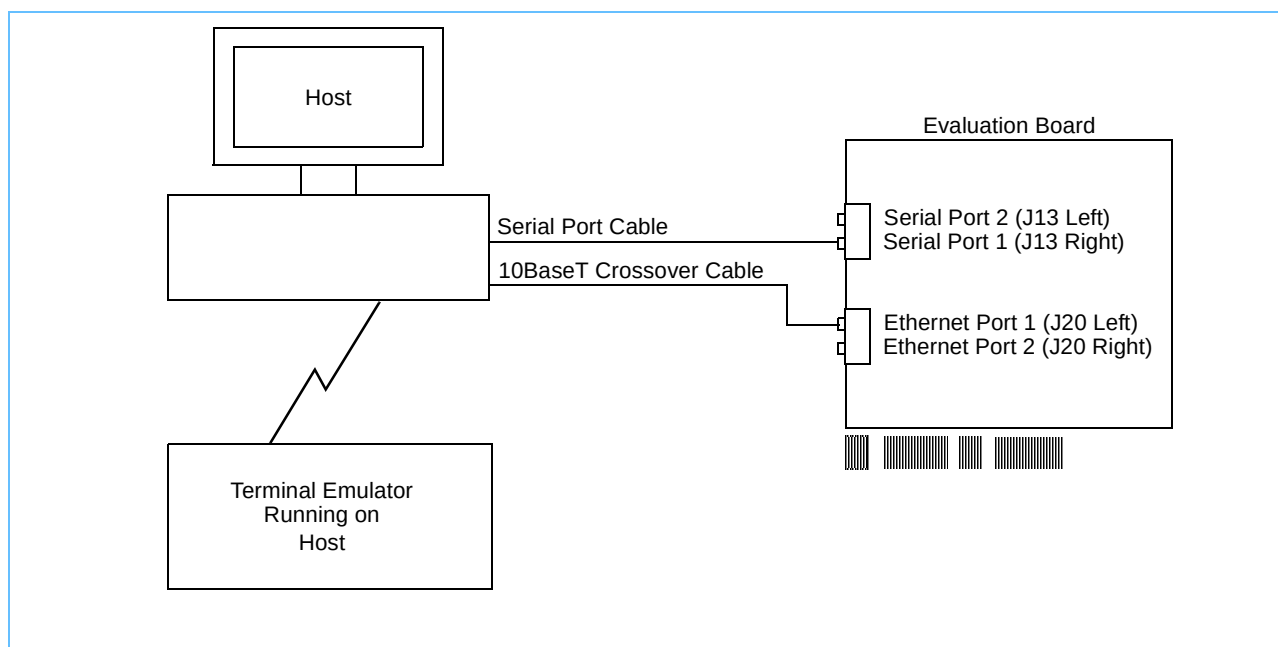
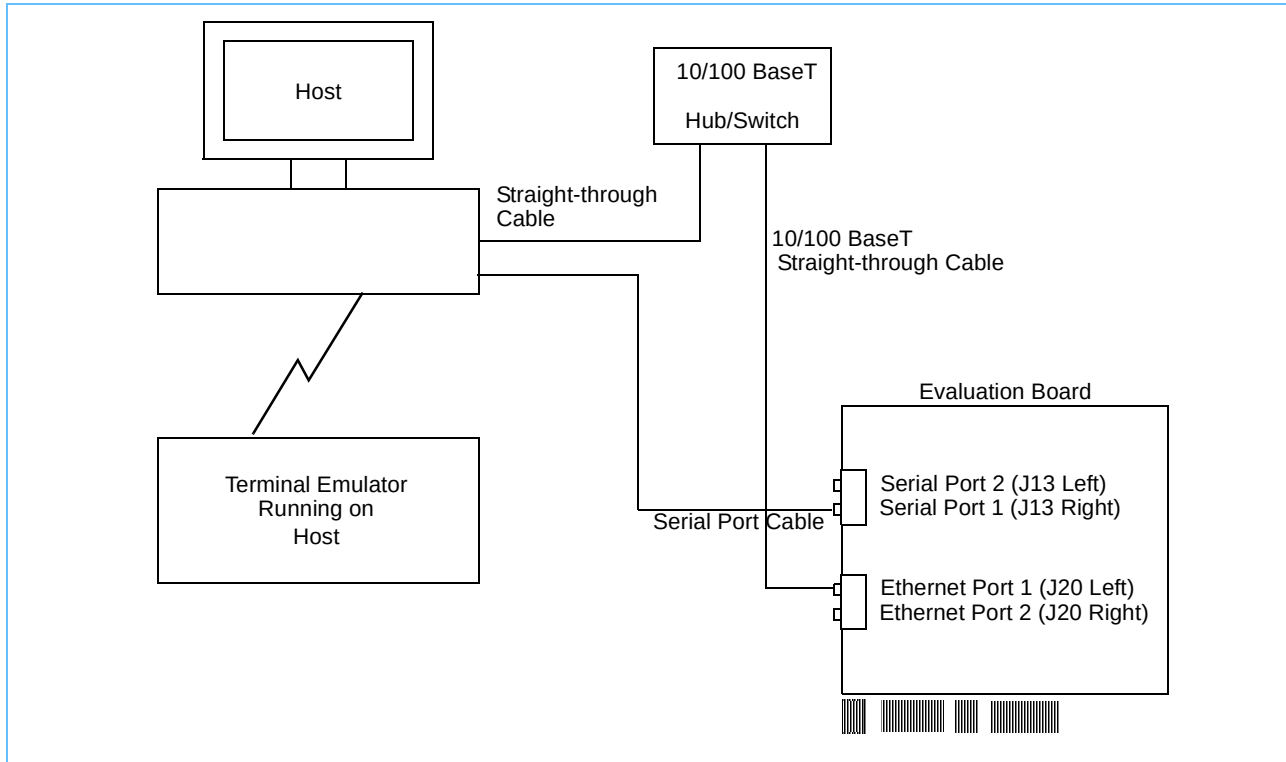


Figure 5-4 shows an Ethernet connection using a hub.

Figure 5-4. Ethernet Connection with Hub



If the connection is to be made to an existing Ethernet network, users should consult their network administrator to insure proper connectivity.

Note: For additional information on location, type, function, and pin assignment for all board connections see *PowerPC 750FX Evaluation Board User's Manual*.

5.2 Using a Terminal Emulator

The PowerPC Initialization Boot Software (PIBS) transmits/receives data through Serial Port 1 (J13 Right) on the board. Access to PIBS can be achieved by connecting a VT100 (or compatible) terminal directly to Serial Port 1 on the board or by using a terminal emulator running on the host. When using a terminal emulator, access is obtained using a null-modem cable connecting Serial Port 1 on the board to an available serial (or COM) port on the host system. The terminal emulator session should be set for 115200 baud, 8 data bits, no parity, 1 stop bit, and Xon/Xoff flow control. If the serial port 2 is connected to the host computer the terminal emulator session should also be set for 115200 baud, 8 data bits, no parity, 1 stop bit, and Xon/Xoff flow control.

5.2.1 Terminal Emulation

Once all the host-to-board connections have been properly made and power has been supplied to the board, the Windows HyperTerminal program can be used as a terminal emulator to support communications with the PIBS monitor. The steps for setting up the terminal emulator connected to COM1 are as follows:

PPC750FX Evaluation Board Kit

1. Select **Start** from the Windows task bar.
2. Select **Programs**.
3. Select **Accessories**.
4. Select **HyperTerminal**.
5. If you see a window that says "You need to install a modem before you can make a connection. Would you like to do this now?" click on "No". You do *not* need a modem to connect to the board.
6. Select the **Hypertrm** icon.
7. Enter a name, for example "PPCEVB750FX" and select an icon.
8. Select the following:
 - Connect using **Direct to Com 1**(default)
 - Bits per second – **115200**
 - Data bits – **8** (default)
 - Parity – **None** (default)
 - Stop Bits – **1** (default)
 - Flow Control – **Xon/Xoff**
9. Select **OK**.

After resetting the board, the PIBS shell prompt should appear in the HyperTerminal window. If it does not, check your HyperTerminal settings and ensure proper connectivity between the host and the board.

5.3 Board Reset

When the connectors have been installed and power is applied to the board, press the board's On/Off switch to power up the board. Pressing the Reset switch causes the processor and the communications controllers to reset. After PIBS (resident in FLASH) initializes the processor and board peripherals, the PIBS shell prompt is displayed if a properly configured terminal emulator is attached to Serial Port 1 (J13 Right) on the board. Details of PIBS operation are provided in *PowerPC Initialization Boot Software (PIBS)* on page 31.



Part II. PowerPC Initialization Boot Software (PIBS)



6. PIBS Overview

The PowerPC Initialization Boot Software (PIBS) is programmed in the 1MB boot FLASH memory on the evaluation board. PIBS initializes the CPU0 PPC750FX microprocessor and system level board components, and includes support for loading, debugging, and executing application programs. PIBS also provides a means for executing programs independently on CPU1.

All user interaction with PIBS occurs through a serial port connection between the board and the host system. The host serial port must be configured for 115200 baud, 8 data bits, no parity, one stop bit, and Xon/Xoff flow control. The null-modem cable included with the evaluation kit should be used to connect Serial Port 1 on the board (J13 Right) to an available serial port on the host system. Once connected and configured properly, a terminal emulator session running on the host allows the user to interact with the PIBS shell.

PIBS supports host-to-board file transfer using a serial port or Ethernet network connection. PIBS includes a Debug Monitor that allows source-level application debug over Ethernet using the RISCWatch debugger running on the host. This debug capability is not supported over a serial port connection.

PIBS for the PPC750FX platform consists of a single image called PIBS2. The source code for PIBS is included in the Board Support Package of the evaluation board kit software. The code can be modified, re-compiled, and re-programmed into the FLASH memory on the board.

6.1 PPC750FX External Hardware Support

PIBS includes support for the on-board SDRAM, SRAM, and FLASH memories, as well as the on-board serial and Ethernet ports. Each Ethernet is assigned a unique hardware address that is displayed, along with other system information, after a successful PIBS boot. The PIBS ent0 and ent1 Ethernet interfaces correspond to the Ethernet Port 1 (J20 Left) and Ethernet Port 2 (J20 Right) RJ45 connectors on the board, respectively.

Note: Ethernet hardware addresses can be set using the PIBS `set hwdaddr0=your-hardware-address` command. See `hwdaddr0:hwdaddr1` on page 64 for more details. After the address is set, the board must be reset so that the Ethernet controller can be programmed with the correct hardware address. The ethernet hardware addresses are set at the factory and should not be changed by the user.

6.2 Quick Start

Connect the PPC750FX evaluation board Serial Port 1 (J13 Right) to the PC using the null-modem cable included with the kit. On the host computer, start up the terminal emulation software. On a Windows PC, Hyperterminal can be used as a terminal emulator. The required serial port connection settings are 115200 baud, 8 data bits, no parity, one stop bit, and Xon/Xoff flow control. Once the terminal emulation software package is up and running, power up the platform. At this point, the PIBS shell prompt should appear on the screen. If it doesn't, verify the terminal emulation software configuration and make sure the platform is powered on. Once the PIBS shell appears, all PIBS shell commands can be issued. The PIBS commands are explained in *Section 7 PIBS Commands* on page 35.

PPC750FX Evaluation Board Kit

After connecting the board to an existing Ethernet network or using the supplied Ethernet cross-over cable to connect the board directly to a host system, the PIBS `ifconfig` command (page 48) can be used to configure the Ethernet network interface. To set the `ifconfig` network configuration data automatically after a reboot, set the PIBS `ifconfigcmd` variable (page 65) accordingly. The PIBS `ping` command (page 49) can be used to test Ethernet connectivity between the host and the board.

6.3 PIBS Memory Map

The PIBS2 binary is programmed into the FLASH memory on the board. After reset, PIBS executes from FLASH, initializes the processor, memory controller, and memory, and then copies itself to RAM for faster execution. *Table 6-1* describes the PIBS memory map for the PPC750FX platform.

Table 6-1. PIBS Memory Map

0xFFFFFFFF - 0xFFFF0000	PIBS data
0xFFFEFFFF - 0xFFF02000	PIBS2 in FLASH
0xFFF01FFF - 0xFFF00000	High Vectors
0x0FFFFFFF - 0x00A00000	Unused SDRAM
0x009FFFFF - 0x00600000	CPU1 application space
0x005FFFFF - 0x00200000	CPU0 application space
0x001FFFFF - 0x00002000	PIBS2 text, data, stack, RAM
0x00001FFF - 0x00000000	Low Vectors

7. PIBS Commands

This section explains the commands available to the user from the PIBS shell. The following explains the syntax used in the examples.

- `Courier` font indicates text typed in on the PIBS command line.
- *Italic* text indicates a general name substitution.
- **Bold** text indicates a command or variable name or a file name.
- Text of the form (option1|option2|option3) indicates the user must type in one and only one of the options.
- Text of the form (*general_text*)+ indicates the *general_text* token must be replicated at least once and possibly multiple times.
- The text <ENTER> means either the ENTER or RETURN key on the computer keyboard is pressed.

Note: Addresses used in the PIBS command descriptions are for example only and may not be valid for all evaluation board platforms.

alias

Command

PPC750FX Evaluation Board Kit User's Manual



Synopsis

Assigns another name to a PIBS command

Syntax

```
alias alias_name=system_command
```

Parameters

alias_name A name to be used as an alias.

system_command Name of a system command to be aliased

Example

```
alias my_help=help <ENTER>
```

Result: typing in **my_help** on the command line now brings up the help menu.

Notes

None

Synopsis

Displays system information and settings

Syntax

`boardinfo`

Parameters

None

Example

`boardinfo <ENTER>`

Result: Displays the current system settings. Depending on the PIBS version and the hardware platform, the results shown here may be different than what is actually displayed.

```
board config data version: 1.1
processor name           : 750FX
processor PVR value      : 0x70000202
total SDRAM memory      : 268435456
system clk period (ps)  : 7500
system clk frequency (Hz): 133333333
PLL0 strap frequency    : 733333331
PLL0 configuration      : externally strapped
PLL0 frequency int config: 0
PLL1 frequency int config: 0
Active PLL (0 or 1)     : 0
CPU frequency           : 733333331
CPU frequency ind. method: 733325992
timer clock frequency   : 33333333
serial clk frequency    : 3686400
processor number        : 0
bridge init vector (LO) : 201f1162
bridge init vector (HI) : 00150560
HID0 value              : 0000c024
HID1 value              : 58000000
CPLD revision level     : 18
Board revision level     : 01
Ethernet hardware addr 0 : 0004ACE323DC
Ethernet hardware addr 1 : 0004ACE323DD
```

Notes

Table 7-1 explains the meaning of the fields.

Table 7-1. Board Configuration Information

Variable	Board config field description
board config data version	The current version of the board configuration data structure.
processor name	The name of the PowerPC processor used in the specific platform.
processor PVR value	The PVR register contents of the PowerPC processor.
total SDRAM memory	The amount of SDRAM or DDR-SDRAM memory installed in the unit.
system clk period (ps)	The PowerPC chip input system clock period displayed in picoseconds.
system clk frequency (Hz)	The PowerPC chip input system clock frequency displayed in Hertz.
PLL0 strap frequency	The PLL0 frequency, in Hertz, externally configured using board switches that drive the chip's PLL strapping pins. This PLL frequency is used during board reset.
PLL0 configuration	The current configuration of the PLL0.
PLL0 frequency int config	The PLL0 frequency, in Hertz, internally configured using software to vary the processor speed.
PLL1 frequency int config	The PLL1 frequency, in Hertz, internally configured using software to vary the processor speed.
Active PLL (0 or 1)	Show the currently active PLL.
CPU frequency	The calculated CPU core frequency, in Hertz, based on the input system clock frequency and PowerPC chip PLL strapping.
CPU frequency ind. method	The CPU core frequency as indicated by detecting the number of instructions executed during a specific period of time. This can be used to verify the validity of the CPU core frequency.
timer clock frequency	The frequency, in Hertz, of the CPU core timebase.
serial clk frequency	The frequency, in Hertz, of the serial port clock.
processor number	For multiprocessor systems, shows which processor PIBS is running on.
bridge init vector (LO)	Value of the bridge chip "reset sample low register".
bridge init vector (HI)	Value of the bridge chip "reset sample high register".
HID0 value	The HID0 register value for the processor being used.
HID1 value	The HID1 register value for the processor being used.
CPLD revision level	The revision level of the code used to program the CPLD (Complex Programmable Logic Device) on the board.
Board revision level	Shows current board revision level
Ethernet hardware addr 0	The hardware address assigned to the Ethernet 0 port on the board.
Ethernet hardware addr 1	The hardware address assigned to the Ethernet 1 port on the board.

Synopsis

Bootfile command is responsible for file loading, relocating and execution. File formats supported are the IBM image format, ELF file and binary file formats. See *IBM Image File Format Description* on page 75 for specifics on the IBM image format. ELF and IBM image formats are relocated by PIBS as indicated by their headers before execution begins. PIBS does not relocate a file stored in the binary file format. With a binary file, execution begins at the *addr* specified. If program relocation is necessary for a binary file, it is the program's responsibility to relocate itself.

Before PIBS branches to the user file, the system is put into a state such that system components cannot interfere with user program execution. This includes the following:

1. Data address translation is enabled. MSR.DR = 0
2. Instruction address translation is enabled. MSR.IR = 0
3. Machine check exceptions are enabled. MSR.ME = 0
4. External interrupts are disabled at the interrupt controller level.

PIBS also performs the following steps before branching to the user file.

1. The data cache is completely flushed.
2. The instruction cache is invalidated.

Before execution of the user file starts, two parameters are passed in through the processor's general purpose registers. These parameters point to data structures residing in RAM, so the user program must not overwrite them unless they are not needed anymore.

1. GPR 3: A pointer to the user supplied program information, such as boot parameters, located in the PIBS environment variable *userdata*. See *userdata* on page 69 for more information on the *userdata* environment variable.
2. GPR 4: A pointer to the board configuration structure which contains system information such as CPU speed, various bus frequencies and total amount of memory installed. See page 37 for more information.

Syntax

```
bootfile (flash|flashb|eth|ethl|ethd|ethl2|serial|serial-  
all|seriall2|ram) [addr]
```

Parameters

flash

Relocate and execute an IBM image or ELF file residing in FLASH. If the *addr* option is specified, PIBS tries to locate the IBM image or ELF file at that address. The **bootfile** command fails if the IBM image or ELF file at the specified address is invalid. If the *addr* option is not used, PIBS uses the **imgaddr** environment value if it is valid. If **imgaddr** isn't set to a valid address, PIBS searches through the FLASH and relocates/executes the first valid IBM image or ELF file found. If no valid IBM image or ELF file is found, the **bootfile** command fails.

<i>flashb</i>	Relocate and execute an IBM image or ELF file, or execute a binary file, residing in FLASH. If the <i>addr</i> option is specified, PIBS tries to locate a IBM image or ELF file at that address. If an IBM image or ELF file is not located at that address, PIBS treats the file as a binary file and starts program execution by branching to the specified address. If the <i>addr</i> option is not used, PIBS uses the imgaddr environment value instead if it is valid. If imgaddr isn't set to a valid address, PIBS searches through the FLASH and branches to the first valid IBM image or ELF file found. If a IBM image or ELF file is located, then the <i>flashb</i> option behaves just like the <i>flash</i> option. The <i>flashb</i> option behaves differently only if the file is not a IBM image or an ELF file.
<i>eth</i>	Load, relocate and execute an IBM image or ELF file across the network. If using the <i>eth</i> option, the <i>addr</i> option is not used. The <i>eth</i> , <i>ethl</i> , <i>ethl2</i> and <i>ethd</i> options require a TFTP server/daemon running on the host computer that supplies the user program. See <i>Configuring TFTP</i> on page 74 for more information on TFTP server system requirements and installation. PIBS environment variables ipdstaddr0 and bootfilename supply the host computer's IP address and directory path and filename information respectively.
<i>ethl</i>	Similar to the <i>eth</i> option but only performs the load step to get the IBM image or ELF file into memory. Program relocation and execution does not take place.
<i>ethl2</i>	Similar to the <i>ethl</i> option but used to load code image for CPU1.
<i>ethd</i>	Similar to the <i>eth</i> option but adds debug output messages.
<i>serial</i>	Load, relocate and execute an IBM image or ELF file using the serial port as the transport medium and kermit as the file transfer protocol. If using the serial option, the <i>addr</i> option is not used.
<i>seriall</i>	Similar to the <i>serial</i> option but only performs the load step to get the IBM image or ELF file into memory. Program relocation and execution does not take place.
<i>seriall2</i>	Similar to the <i>seriall</i> but used to load code image for CPU1.
<i>ram</i>	If bootfile command was previously executed with either the <i>ethl</i> or <i>seriall</i> option to get the IBM image or ELF file loaded into memory, calling bootfile with the <i>ram</i> option starts program relocation and execution. If the <i>addr</i> option is used it specifies the start address of the loaded file. If the <i>addr</i> option is not used the default RAM location is used instead.
<i>addr</i>	Used with the <i>flash</i> , <i>flashb</i> or <i>ram</i> option and specifies where to look in the FLASH or RAM for the IBM image, ELF or binary file.

Example

```
bootfile flash 0xfD000000 <ENTER>
```

Result: relocate/execute an IBM image or ELF file from FLASH address 0xFD000000

```
bootfile flash <ENTER>
```

Result: If the **imgaddr** environment variable is set, the **bootfile** command uses it as the IBM image/Elf file starting address. If the **imgaddr** environment variable is not set, PIBS searches through the FLASH and relocate/execute the first IBM image or ELF file found.

```
bootfile flashb 0xFD000000 <ENTER>
```

Result: First, PIBS determines if the file at 0xFD000000 is a IBM image or ELF file. If so, the file is relocated if necessary and executed. Otherwise, PIBS assumes it is a binary file and branches to address 0xFD000000.

```
bootfile flashb <ENTER>
```

Result: execute a binary file or relocate/execute a IBM image or ELF file from FLASH address specified by the PIBS **imgaddr** environment variable. If the **imgaddr** environment variable setting is valid, PIBS determines if the file at the **imgaddr** start address is a IBM image or ELF file. If so, the file is appropriately relocated and executed. Otherwise, PIBS assumes it is a binary file and branches to the address defined by **imgaddr**. If the **imgaddr** environment variable address setting is invalid, the **bootfile** command relocates/executes the first IBM image or ELF file found in FLASH storage. If no IBM image or ELF file is found, the command fails.

```
bootfile eth <ENTER>
```

Result: load a IBM image or ELF file from the network and once loaded, perform relocation and branch to the program entry point. PIBS environment variables **ipdstaddr0** and **bootfilename** supply the host computer's IP address and directory path/filename respectively. The host computer must be running a TFTP server/daemon and have access to the file at the specified directory path.

```
bootfile serial <ENTER>
```

Result: load an IBM image or ELF file using the serial port kermit file transfer protocol and once loaded, perform relocation and start execution.

```
bootfile ethl2 <ENTER>
```

Result: load an IBM image or ELF file from the network and once loaded, perform relocation. PIBS environment variables **ipdstaddr0** and **bootfilename** supply the host computer's IP address and directory path/filename respectively. The host computer must be running a TFTP server/daemon and have access to the file at the specified directory path. The loaded file will be executed on CPU1 after an image is loaded on CPU0. See *Section 15 MultiProcessor Sample* on page 87 for additional details.

Notes

None

Synopsis

Sets the processor clocking source and/or frequency.

Syntax

```
chipclk (pll0|pll1|default) [frequency] [imm]
```

Parameters

<i>pll0</i>	If the frequency parameter is specified, software internally configures PLL0 for the desired processor frequency if it is not the active PLL. This setting becomes active only after a subsequent <code>chipclk</code> command is issued for <code>pll0</code> without the frequency parameter specified. If the <code>imm</code> parameter is specified the setting becomes active immediately.
<i>pll1</i>	If the frequency parameter is specified, software internally configures PLL1 for the desired processor frequency if it is not the active PLL. This setting becomes active only after a subsequent <code>chipclk</code> command is issued for <code>pll1</code> without the frequency parameter specified. If the <code>imm</code> parameter is specified the setting becomes active immediately.
<i>default</i>	Uses the external configuration and range strapping pins to control PLL0 and set the processor frequency. The frequency parameter need not be specified since the frequency will be set based on external configuration strapping. The <code>default</code> option is valid only when PLL1 is the active PLL.
<i>frequency</i>	Desired processor frequency in MHz. Specifying a value of "off" will disable the specified PLL if it is not the active clocking source. This can be used to conserve power.
<i>imm</i>	Causes the specified setting to become active immediately

Example

```
chipclk pll1 400 <ENTER>
```

Result: software internally configures PLL1 for 400MHz if PLL1 is not the active PLL.

```
chipclk pll1 <ENTER>
```

Result: PLL1 is made the active PLL and the processor frequency is now 400MHz.

```
chipclk pll0 off <ENTER>
```

Result: PLL0 is disabled to conserve power since it is no longer the active PLL.

```
chipclk default <ENTER>
```

Result: PLL0 is made the active PLL and external configuration set by strapping is used to set the processor frequency.

Note

For the PPC750FX evaluation board, only processor frequencies between 400MHz and 733MHz are supported. This corresponds to values of 400 to 733 for the frequency parameter of the **chipclk** command since it is specified in MHz. Users should be aware that some frequency values may not be accepted due to multiplier constraints and dependencies. See the PowerPC 750FX Data Sheet for additional information regarding the PLL usage and limitations.

Synopsis

Displays memory contents as translated by the MMU. Memory contents are displayed in their hexadecimal value as well as their ASCII character value.

Syntax

```
display mem_addr [num_bytes]
```

Parameters

<i>mem_addr</i>	The 32 bit effective start address. This value can either be a 32 bit decimal or hexadecimal value. Hexadecimal values are preceded by a 0x specifier.
<i>num_bytes</i>	The number of memory bytes to display starting with the start address defined by the <i>mem_addr</i> option. This value is rounded up to the nearest value evenly divisible by 4. For example, if 89 bytes are requested, 92 will be displayed. If this option isn't specified on the command line, 16 bytes is the default value used.

Example

```
display 300 99
```

result: Displays 100 bytes starting at address 0x12c (300 decimal).

```
0000012c:0000012c 00000130 00000134 00000138 | ..., ...0 ...4 ...8 |
0000013c:0000013c 00000140 00000144 00000148 | ...< ...@ ...D ...H |
0000014c:0000014c 00000150 00000154 00000158 | ...L ...P ...T ...X |
0000015c:0000015c 00000160 00000164 00000168 | ... \ ... ` ...d ...h |
0000016c:0000016c 00000170 00000174 00000178 | ...l ...p ...t ...x |
0000017c:0000017c 00000180 00000184 00000188 | ...| .... .... |
0000018c:0000018c                                     | .... |
```

```
display 0x300 101
```

result: Displays 104 bytes starting at address 0x300

```
00000300:00000300 00000304 00000308 0000030c | .... .... .... |
00000310:00000310 00000314 00000318 0000031c | .... .... .... |
00000320:00000320 00000324 00000328 0000032c | ... ..$ ...( ..., |
00000330:00000330 00000334 00000338 0000033c | ...0 ...4 ...8 ...< |
00000340:00000340 00000344 00000348 0000034c | ...@ ...D ...H ...L |
00000350:00000350 00000354 00000358 0000035c | ...P ...T ...X ... \ |
00000360:00000360 00000364                                     | ... ` ...d |
```

Notes

None

Synopsis

Displays the value of a PIBS variable or echoes text back to the display.

Syntax

```
echo ($variablename|text)
```

Parameters

<i>variablename</i>	A valid name of a PIBS system variable. See <i>PIBS Environment Variables</i> on page 57 for valid PIBS system variable names and their descriptions.
<i>text</i>	Any generic text to get outputted back to the display.

Example

```
echo $dhcp0 <ENTER>
```

Result: The value of the *dhcp0* environment variable is outputted to the display. For example, if the *dhcp0* environment variable is set to `true`, then “true” appears on the display.

```
echo $bootfilename <ENTER>
```

Result: The value of the *bootfilename* environment variable appears on the display

```
echo generic text <ENTER>
```

Result: “generic text” appears on the display

Notes

None

help

Command

PPC750FX Evaluation Board Kit User's Manual



Synopsis

Displays the help screen.

Syntax

help [command_name]

Parameters

command_name If provided, displays help information for *command_name*.

Example

help <ENTER>

Result: displays and gives a brief description of the available PIBS commands. Depending on the PIBS version level and the hardware platform, the help results shown here may be different than what is actually displayed.

Very simple shell for PIBS can be used to:

1. Execute following commands:

- alias : assigns an alias to a command
- boardinfo : displays board information
- bootfile : load file and jumps to files entry point
- chipclk : sets chip clocking information
- echo : displays value of the PIBS variable
- help : displays help information
- ifconfig : configures/displays network interface info
- ping : sends an echo request to a network host
- reset : resets the board
- route : manipulates routing tables
- set : sets/displays value of the PIBS variable
- storefile : stores file in flash
- unalias : removes command alias
- version : displays PIBS version string

2. Set and display PIBS variables. PIBS variable values are preserved across reboots. PIBS variables are: autoboot, autobootdelay, bootfilename, chipclkcmd, dhcp0, dhcp1, hwdaddr0, hwdaddr1, ifconfigcmd0, ifconfigcmd1, imgaddr, ipdstaddr0, routecmd, userdata. PIBS variables are assigned using the set command, and displayed using echo command.

3. Display command help. For example: help bootfile.

help alias <ENTER>

Result: gives a more detailed description for the **alias** command.



Notes

Help usage displays for the **ping**, **ifconfig**, and **route** commands are not consistent with the PIBS shell syntax but rather with the EPOS shell syntax. When issuing these commands from the PIBS shell they should be used as documented in *Section 7 PIBS Commands*.

Synopsis

Configures the network interface

Syntax

```
ifconfig interf [af] [addr [dest_ad]] [up] [down] [netmask]  
[metric] [trail|-trail] [arp|-arp]
```

Parameters

<i>interf</i>	The interface to configure (ent0 or ent1)
<i>af</i>	The address family
<i>addr</i>	The host address
<i>dest_ad</i>	The destination address for that interface
<i>up</i>	Bring <i>interf</i> up
<i>down</i>	Bring <i>interf</i> down
<i>netmask</i>	Specifies network mask
<i>metric</i>	Number of hops to destination
<i>trail</i>	Indicates use of trailers
<i>arp</i>	Indicates use of arp

Example

```
ifconfig ent0 up <ENTER>
```

Result: brings the interface up

```
ifconfig ent0 down <ENTER>
```

Result: brings the interface down

```
ifconfig ent0 192.168.0.1 up <ENTER>
```

Result: sets the TCP/IP address to 192.168.0.1 and brings the interface up

```
ifconfig ent0
```

Result: displays the addressing information for the ent0 interface

Notes

When using DHCP to provide the IP information, do not use the **ifconfig** command to change system settings. Valid interface names are ent0 and ent1.

Synopsis

Sends an icmp echo request to a network host.

Syntax

```
ping [-rv] host [size] [packets]
```

Parameters

<i>r</i>	Bypass the normal routing tables
<i>v</i>	Verbose output
<i>host</i>	IP address destination
<i>size</i>	The size in bytes of the data portion of the ping packet.
<i>packets</i>	The number of echo request packets to send.

Example

```
ping 192.168.0.1 256 4 <ENTER>
```

Result: transmits four icmp ping packets to IP address 192.168.0.1. Each ping packet includes a 256 byte data section.

Notes

If the *size* parameter is not specified, the data portion of the ping packet defaults to 64 bytes. If the *packets* parameter is not specified, the **ping** command defaults to sending out three icmp echo requests.

reset

Command

PPC750FX Evaluation Board Kit User's Manual



Synopsis

Resets the board causing a reboot of the software

Syntax

reset

Parameters

None

Example

reset

Result: resets the entire board including the processors and onboard peripherals.

Notes

None

Synopsis

Manipulates routing tables

Syntax

```
route (add|delete) [net|host] (dest|default) router [metric]
```

Parameters

<i>add</i>	Add a route
<i>delete</i>	Delete a route
<i>net</i>	Specifies a network route
<i>host</i>	Specifies a host route
<i>dest</i>	The destination host or network
<i>default</i>	All destinations not defined with another entry
<i>router</i>	The next-hop in the path to the destination
<i>metric</i>	Number of hops to destination (required for add)

Example

```
route add net default 192.168.0.40 10
```

Result: IP address 192.168.0.40 is added as a default route to the router table with a maximum of 10 hops to the destination.

```
route delete net default 192.168.0.40
```

Result: delete router entry 192.168.0.40.

```
route delete net default 0
```

Result: delete all router entries.

Notes

None

set

Command

PPC750FX Evaluation Board Kit User's Manual



Synopsis

Assigns a value to a system environment variable or displays the current system variables and their values.

Syntax #1

```
set [pibs_variable=value]
```

Parameters

pibs_variable Name of the variable in PIBS to use. Type in **set** <ENTER> on the command line for a complete list.

value Value to set the variable to

Example

```
set dhcp0=true <ENTER>
```

Result: the next time the board is booted, PIBS will start up the dhcp client service.

```
set <ENTER>
```

Result: the current setting of all the PIBS variables is outputted to the display

Notes

Depending on the PIBS version level and the hardware platform, the environment variable name in the above example may be different than what is actually used.

Synopsis

Stores a file into system FLASH. A file may either be a IBM image, ELF or raw binary file.

Syntax

```
storefile (bin | pibs ) (eth | ethd | serial) [addr]
```

Parameters

<i>bin</i>	Takes a binary, IBM image or ELF file and stores it into FLASH. When using the <i>bin</i> parameter, the <i>addr</i> option must be specified.
<i>pibs</i>	Used to reprogram PIBS2 into the FLASH. The <i>addr</i> option is ignored with this parameter. When using the <i>pibs</i> option, the source PIBS2 file must be a binary file.
<i>eth</i>	Load the file from another host over the network. The <i>eth</i> option requires a TFTP server/daemon running on the host computer that supplies the user program. See <i>Configuring TFTP</i> on page 74 for more information on TFTP server system requirements and installation. PIBS environment variables ipdstaddr0 and bootfilename supply the host computer's IP address and directory path/filename.
<i>ethd</i>	Similar to the <i>eth</i> option but adds debug output messages.
<i>serial</i>	Load the file over the serial port using kermit file transfer protocol.
<i>addr</i>	Address to place the loaded file into FLASH. This option must be specified when using the <i>bin</i> parameter.

Example

```
storefile bin eth 0xFD000000 <ENTER>
```

Result: Load in a file over the network and store it into FLASH at address 0xFD000000.

```
storefile bin serial 0xFD000000 <ENTER>
```

Result: Load in a file over the serial port and store it into FLASH at address 0xFD000000.

```
storefile pibs eth <ENTER>
```

Result: Load in PIBS2 over the network and store it into FLASH. A reboot soon follows.

```
storefile pibs serial <ENTER>
```

Result: Load in PIBS2 over the serial port and store it into FLASH. A reboot soon follows.



Notes

None

Synopsis

Removes an alias to a command

Syntax

```
unalias alias_name
```

Parameters

alias_name Name of the alias to remove from the system

Example

```
unalias my_help <ENTER>
```

Result: If **my_help** was an alias to another command it no longer works.

Notes

None

version

Command

PPC750FX Evaluation Board Kit User's Manual



Synopsis

Displays the PIBS version string

Syntax

`version`

Parameters

None

Example

`version <ENTER>`

Result: The PIBS version string is outputted to the display.

Notes

None

8. PIBS Environment Variables

The PIBS environment variables fall under two categories. *Table 8-1* lists variables that control PIBS system initialization before the shell comes up, while *Table 8-2* lists variables that indirectly control the function of various PIBS shell commands.

Table 8-1. PIBS System Initialization Environment Variables

Variable name	Variable description
autoboot	Controls automatic program loading on board bootup
autobootdelay	Controls the auto boot delay
chipclkcmd	Controls chip clocking
dhcp0	Enables/disables DHCP support on Ethernet 0
dhcp1	Enables/disables DHCP support on Ethernet 1
hwdaddr0	The MAC hardware address for Ethernet 0
hwdaddr1	The MAC hardware address for Ethernet 1
ifconfigcmd0	Controls Ethernet device initialization during PIBS boot
ifconfigcmd1	Controls Ethernet device initialization during PIBS boot
routeclmd	Controls routing on board bootup

Table 8-2. PIBS Runtime Environment Variables

Variable name	Variable description
aliaslist	Allows users to supply alternate names for the PIBS commands.
bootfilename	Contains the path and filename of the boot file residing on the host computer. Used in conjunction with TFTP on the host computer.
imgaddr	Contains the FLASH address of the boot file.
ipdstaddr0	Contains the host computer's IP address containing the boot file. Used in conjunction with TFTP on the host computer.
userdata	Gives a place in FLASH to store user supplied program information.

aliaslist

Environment Variable

PPC750FX Evaluation Board Kit User's Manual



Synopsis

The **aliaslist** variable allows users to supply alternate names for PIBS commands.

Syntax

```
set aliaslist=(alias_name=cmd_name;)+
```

Parameters

<i>alias_name</i>	Name of the alias to use
<i>cmd_name</i>	Name of the PIBS command to set the alias to

Usage

runtime

Example

```
set aliaslist=h=help;v=version;
```

Result: Typing h <ENTER> on the command line initiates the **help** command. Typing v <ENTER> on the command line initiates the **version** command.

Notes

Using the **alias** and **unalias** PIBS commands indirectly modifies the **aliaslist** variable.

Synopsis

The **autoboot** variable contains the parameters for the **bootfile** command. After system initialization, PIBS can be instructed to automatically issue a **bootfile** command. Automatic boot is enabled by placing correct bootfile parameters in the **autoboot** variable. Automatic boot is disabled by clearing the **autoboot** variable. The **autoboot** variable can be cleared by executing the command

```
set autoboot=<ENTER>
```

Syntax

```
set autoboot=bootfile_parms
```

Parameters

<i>bootfile_parms</i>	bootfile command parameters to be issued to the bootfile command. For more information on the bootfile command, see <i>bootfile</i> on page 39.
-----------------------	--

Usage

boottime

Example

```
set autoboot=flashb 0xFD000000 <ENTER>
```

Result: after system boot, PIBS executes a

```
bootfile flashb 0xFD000000 <ENTER>
```

command. Notice that the **bootfile** command name should not be specified in the **autoboot** variable.

Notes

The **autoboot** variable is used in conjunction with the **autobootdelay** variable.

Synopsis

If the autoboot function is enabled (as defined on page 59), **autobootdelay** variable sets the amount of delay from after the system is initialized to the time when the autoboot operation begins. This enables the user to abort the autoboot process.

Syntax

```
set autobootdelay=time
```

Parameters

<i>time</i>	Delay time ranging from 1 second to 999 seconds. Any time specified out of this range causes the autobootdelay to default to 10 seconds.
-------------	---

Usage

boottime

Example

```
set autobootdelay=10 <ENTER>
```

Result: After the next system reboot, if autoboot function is enabled, the **bootfile** command specified in the **autoboot** variable will be executed after a 10 second delay unless the user presses a key on the console.

Notes

None

Synopsis

The **bootfilename** variable holds the path and filename of the boot file residing on the host computer.

Syntax

```
set bootfilename=file_location
```

Parameters

<i>file_location</i>	Path and filename of the boot file
----------------------	------------------------------------

Usage

runtime

Example

```
set bootfilename=c:\images\bootfile.img <ENTER>
```

Notes

The **bootfile** and **storefile** command use the PIBS **bootfilename** environment variable in conjunction with the **ipdstaddr0** variable to locate the boot file on the host computer using TFTP. See *Configuring TFTP* on page 74 for more information on how to configure TFTP on the host computer.

Synopsis

The **chipclkcmd** variable contains the bootup parameters for the chipclk command. After system initialization, PIBS can be instructed to automatically issue a **chipclk** command with the supplied parameters in the **chipclkcmd** environment variable. This can be used to modify the processor's clocking source and frequency.

Syntax

```
set chipclkcmd=chipclk_parms
```

Parameters

<i>chipclk_parms</i>	chipclk command parameters to be issued to the chipclk command. For more information on the chipclk command, see <i>chipclk</i> on page 42.
----------------------	---

Usage

boottime

Example

```
set chipclkcmd=pll1 400 imm<ENTER>
```

Result: After the next system boot, PIBS executes a `chipclk pll1 400 imm<ENTER>` command to change the processor clocking source to PLL1 and the frequency to 400MHz.

Notes

None

Synopsis

The **dhcp0:dhcp1** variables control DHCP client operation. If value of the dhcp variable is set to **true**, following a reset, PIBS enables the associated DHCP client. If value of the dhcp variable is set to false, DHCP client support for that device will be disabled after rebooting. If DHCP client for a particular device is disabled it is up to the user to configure the Ethernet networking through ifconfig and route commands.

Syntax

```
set dhcp_variable_name=boolean
```

Parameters

dhcp_variable_name dhcp0, dhcp1

boolean true or false

Usage

boottime

Example

```
set dhcp0=true <ENTER>
```

Result: dhcp support for the ent0 interface is enabled on the next reboot.

```
set dhcp0=false <ENTER>
```

Result: dhcp support for the ent0 interface is disabled on the next reboot.

Notes

When using DHCP to provide the IP information, do not use the **ifconfig** command to change system settings for the same Ethernet device. When DHCP is used, PIBS must be rebooted when connecting the platform to a different network.

Synopsis

The **hwdaddr0:hwdaddr1** variables are used to program the Ethernet hardware (MAC) addresses.

Syntax

```
set device_name=mac_addr
```

Parameters

<i>device_name</i>	hwdaddr0, hwdaddr1.
<i>mac_addr</i>	The Ethernet's assigned six digit hexadecimal hardware address. The number must be specified in hexadecimal. See the included example on how to specify the hexadecimal number.

Usage

boottime

Example

```
set hwdaddr0=0123456789ab <ENTER>
```

Result: on the next system reboot, ent0's address will be programmed to 123456789ab.

Notes

The Ethernet hardware (MAC) address must be set in order to use the Ethernet driver in PIBS. Ethernet hardware addresses are set at the factory and should not be changed by the user

Note: The Ethernet hardware (MAC) address numbers are printed on the labels attached to the board.

Synopsis

The **ifconfigcmd0:ifconfigcmd1** variables contain TCP/IP network information

Syntax

```
set ifconfigcmd_name=config_parm
```

Parameters

<i>ifconfigcmd_name</i>	ifconfigcmd0, ifconfigcmd1
<i>config_parm</i>	ifconfig command line parameters

Usage

boottime

Example

```
set ifconfigcmd0=ent0 up 192.168.0.1 <ENTER>
```

Result: after the next system reboot, PIBS brings up interface ent0 with an IP address of 192.168.0.1

```
set ifconfigcmd1=ent1 up 192.168.0.2 <ENTER>
```

Result: after the next system reboot, PIBS brings up interface ent1 with an IP address of 192.168.0.2

Notes

If DHCP client support is enabled for a particular Ethernet device, the **ifconfigcmd0:ifconfigcmd1** variables are not used to configure the same device. Either of the two ifconfigcmd variables can be used to bring up an Ethernet device.

imgaddr

Environment Variable

PPC750FX Evaluation Board Kit User's Manual



Synopsis

The **imgaddr** variable holds the address of the image file stored in FLASH.

Syntax

```
set imgaddr=flash_addr
```

Parameters

<i>flash_addr</i>	Start address in FLASH of the IBM image, ELF or raw binary program.
-------------------	---

Usage

runtime

Example

```
set imgaddr=0xFD000000 <ENTER>
```

Result: Used in conjunction with the **bootfile** command.

Notes

The PIBS **bootfile** command uses the **imgaddr** variable as the FLASH boot file address if one isn't provided as a command line option to the **bootfile** command. See *bootfile* on page 39 for more information on the **bootfile** command.

Synopsis

The **ipdstaddr0** variable holds the host computer's IP address. This variable is used with the following commands:

1. The **bootfile** command (*bootfile* on page 39) with the **eth**, **ethl**, **ethl2** or **ethd** option to specify the IP address of the host computer running the TFTP server/daemon.
2. The **storefile** command (*storefile* on page 53) with the **eth** option to specify the IP address of the host computer running the TFTP server/daemon.

Syntax

```
set ipdstaddr0=host_addr
```

Parameters

<i>host_addr</i>	Host computer IP address
------------------	--------------------------

Usage

runtime

Example

```
set ipdstaddr0=192.168.0.2 <ENTER>
```

Result: the next time the **bootfile** command is executed with the **eth**, **ethl**, **ethl2** or **ethd** option, the host computer at IP address 192.168.0.2 will be queried for the boot file.

Notes

The PIBS **bootfile** and **storefile** commands use the **ipdstaddr0** variable as the TFTP source address. The PIBS **bootfile** and **storefile** commands use the **bootfilename** variable setting as the TFTP source directory path/filename locator. For more information on TFTP host computer setup, see *Configuring TFTP* on page 74.

route cmd

Environment Variable

PPC750FX Evaluation Board Kit User's Manual



Synopsis

The **route cmd** variable contains system routing information.

Syntax

```
set route cmd=route_parm
```

Parameters

route_parm **route** command line parameters.

Usage

boottime

Example

```
set route cmd=add net default 192.0.0.1 10 <ENTER>
```

Result: after the next system reboot, PIBS adds 192.0.0.1 to the route table as a default route for all networks.

Notes

If DHCP client support is enabled, the **route cmd** variable is not used.

Synopsis

The **userdata** variable stores user defined information. For example, operating system boot parameters can be stored in the **userdata** variable and accessed by the operating system during the boot process.

Syntax

```
set userdata=data
```

Parameters

<i>data</i>	Up to 256 data bytes can be assigned to the userdata variable. PIBS stores the data as a string.
-------------	---

Usage

runtime

Example

```
set userdata=my operating system data
```

Result: A pointer to the content of this variable is passed to the user program through general purpose register R3 right before PIBS branches to the start of the user program.

Notes

This variable is meant to be accessed by user supplied programs. PIBS does not use the contents of this variable in any way.

9. PIBS Software Development

This section describes the PIBS software development process.

9.1 Directory Structure

PIBS uses the IBM Embedded PowerPC Operating System (EPOS). Source code for both PIBS and EPOS is provided with the BSP software.

The EPOS kernel code is included in the BSP `install_dir_path\kernel_7xx` directory tree. The EPOS code includes libraries compiled with the GNU for PowerPC compiler. Makefiles and source code (source code for **netlib.a** and **tcpiplib.a** are not included) are also included, allowing the EPOS libraries to be rebuilt if necessary. A makefile options file, **make.opt**, included in the `kernel_7xx\make` directory, can be modified to support different compilers and/or compiler options. Assuming the GNU for PowerPC compiler is being used, the EPOS libraries can be compiled by issuing the "gnumake" command from the `kernel_7xx\lib` directory. If the IBM HighC C++ compiler is being used the **make.opt_7xx.met** file must be renamed to the **make.opt** file and all the EPOS libraries must be recompiled.

The PIBS code is included in the BSP `install_dir_path\ppc750fx` directory tree. The EPOS kernel libraries are required in order to build a PIBS image. The PIBS code includes libraries compiled with the GNU for PowerPC compiler. Makefiles and source code are also included, allowing the PIBS libraries to be rebuilt if necessary. A makefile options file, **make.opt**, included in the `ppc750fx\make` directory, can be modified to support different compilers and/or compiler options. Assuming the GNU for PowerPC compiler is being used, the PIBS libraries can be compiled by issuing the "gnumake" command from the `ppc750fx\lib` directory. If the IBM HighC C++ compiler is being used the **make.opt.met** file must be renamed to the **make.opt** file and all the EPOS libraries must be recompiled.

If a compiler other than the GNU for PowerPC compiler is going to be used, the EPOS and PIBS libraries need to be recompiled with the same compiler. The exception to this are the **tcpiplib.a** and **netlib.a** libraries. The **tcpiplib.a** and **netlib.a** libraries compiled with the GNU for PowerPC compiler can be used with the application programs compiled with the IBM HighC C++ compiler.

Table 9-1 PIBS directory tree on page 70 shows the .ppc750fx directory tree structure.

Table 9-1. PIBS directory tree

Directory Name	Directory descriptions
asynclib	Serial port device driver code
boardlib	Board specific functions
bootlib	PPC750FX initialization, bridge initialization and code setup functions
clocklib	Implementation of the chipclk PIBS command
enetlib	Ethernet device driver
filelib	Implementation of the bootfile and storefile command as well as DHCP code
flashlib	FLASH device driver
include	Include files
iolib	Low level interrupt handling code
lib	Place holder for precompiled libraries
make	Make utility options files

Table 9-1. PIBS directory tree

Directory Name	Directory descriptions
mathlib	Math function required by PIBS
mmulib	PPC750FX page table functions
pcilib	Place holder for the PCI functions (currently not implemented)
pibslib	PIBS shell
ppclib	Low level assembly functions
rw	RISCWatch support files
sample	Sample applications
sdram	SDRAM initialization code
ticklib	Timer tick (decrementer) handler
timelib	Additional time related functions required by PIBS

9.2 Build Tools

The PIBS software can be built using the IBM GNU for PowerPC or the IBM HighC C++ compilers. Additional host tools are also included in the ***install_dir_path\bin*** directory:

- **makedep.exe** - used within Makefiles to create make dependency lists
- **rm.exe** - used within Makefiles to remove files
- **stoimg.exe** - used to convert files from S-record format to IBM image format
- **tftpd.exe** - TFTP server program used for downloading files from the host when using Ethernet

9.3 Makefile Setup

A makefile options files, **make.opt.gcc** or the **make.opt.met**, included in the **ppc750fx\make** directory, can be modified to support different compilers and/or compiler options. The PLATFORM variable in the makefile options file must be set to PPC750FX_EVB to reflect the target platform. The BASE_KERNEL variable in the makefile options file reflects the EPOS file directory location. The LIBRARY_DIR variable in the makefile options file reflects the directory where the compiler was installed. It would need to be modified if the compiler were installed in a directory other than the default install directory.

9.4 PIBS Build Process

The complete PIBS build process is composed of three main steps.

Before building PIBS, make sure the PLATFORM, BASE_KERNEL, and LIBRARY_DIR variables are set appropriately in the **make.opt** file as discussed in *Makefile Setup* on page 71.

- First, build the EPOS kernel libraries, if necessary, using the following commands.
 1. cd ***install_dir_path***
 2. cd **kernel_7xxlib**

PPC750FX Evaluation Board Kit

3. **gnumake** (builds all the kernel libraries)
- Second, build the PIBS libraries if necessary, using the following commands.
 1. cd *install_dir_path*
 2. cd **ppc750fx\lib**
 3. **gnumake** (builds all the PIBS libraries)
- Third, build the PIBS binaries.
 1. cd *install_dir_path*
 2. cd **ppc750fx\sample**
 3. **gnumake pibs2** (in the **sample** directory) is used to build the FLASH executable PIBS2 image and binary files. See *Table 9-2 Files produced from a make pibs2* on page 72 for a list of the files produced.

Table 9-2. Files produced from a make pibs2

Files produced	File descriptions
sample_pibs2.o	intermediate object file
finalopibs2.map	Map file for the PIBS finalopibs2 binary and ELF files
finalopibs2.bin	Binary file containing PIBS2 used to burn into FLASH. This file is used by the RISCWatch JTAG flash_pibs.cmd file to program PIBS2 into system FLASH. This file can also be programmed into the FLASH using the PIBS storefile command with the pibs parameter.
finalopibs2	ELF file containing PIBS2 used to provide RISCWatch debug information if necessary.

4. **gnumake util** (in **sample** directory) is used to build the **finaloutil** executable file used in conjunction with the **flash_pibs.cmd** RISCWatch JTAG command file to program the PIBS binaries into FLASH storage.

Table 9-3. Files produced by make util command

Files produced	File descriptions
flash_pibs.o	intermediate object file
finaloutil.map	Map file for the finaloutil IBM image and ELF files
finaloutil.img	Image file of the finaloutil utility. This file is not used in anyway.
finaloutil	ELF file used by the RISCWatch flash_pibs.cmd file to program the finalopibs1.bin and finalopibs2.bin files into system FLASH.

9.5 Programming PIBS2 Software in FLASH

This section describes different methods that can be used to program PIBS2 into the FLASH memory on the board. Be aware that if PIBS is used to reprogram the FLASH with a modified PIBS2 binary, and the program loaded contains errors that will not allow downloads, the FLASH may be corrupted and rendered useless. In this case RISCWatch JTAG or a FLASH burner will be needed to restore PIBS in FLASH.

1. The PIBS bootfile command can be used to reprogram PIBS2 into the FLASH with the **finalopibs2.bin** file. See *storefile* on page 53 for details.
2. PIBS2 can be initially placed into FLASH using the **flash_pibs.cmd** RISCWatch (version 5.1 or later or RISCWatch is required) command file. Note that a RISCWatch JTAG probe (not included with the kit) is

required to execute the **flash_pibs.cmd** command file. Before running this command file, make sure the serial port is connected and that the host computer is running the terminal emulation software. For the Windows PC environment, Hyperterminal can be used. The terminal emulation settings required are the same ones used to operate the PIBS shell. After establishing the serial port connection, locate the files in *Table 9-4 Files needed to install PIBS onto the PPC750FX platform using flash_pibs.cmd* on page 73, use the RISCWatch **srchpath add** command to add the path where the files exist, and execute the command file via the RISCWatch **exec flash_pibs.cmd** command. After the command file finishes execution, if RISCWatch doesn't automatically go to the "running" mode, type in **run** RISCWatch command and press <ENTER>. At this point, questions will be displayed to the user through the terminal emulation software program. Answer them appropriately with a "Y" for yes or "N" for no to program PIBS into the FLASH.

Table 9-4. Files needed to install PIBS onto the PPC750FX platform using flash_pibs.cmd

File Name	Description
finalopibs2	ELF file containing PIBS2
finalopibs2.bin	Binary file containing PIBS2
finaloutil	ELF file for PIBS2 FLASH utility
flash_pibs.cmd	RISCWatch command file

9.6 Dual Boot Capability

The PPC750FX evaluation board supports booting from either the 1MB socketed or 32MB on-board FLASH memory. Which FLASH is used to boot the processor is determined by configuration switches set on the board. See the *PPC750FX Evaluation Board User's Manual* for additional information.

PIBS, running from the 1MB socketed FLASH, can be used to program boot code into the 32MB on-board FLASH by using the PIBS **storefile** command. This feature allows users to program boot code for another operating system, into the 32MB FLASH while maintaining PIBS functionality in the 1MB FLASH. Board switches can be used to set which FLASH to use for booting, giving the user dual boot capability.

Note: PIBS can only be executed from 1MB socketed FLASH. PIBS execution from 32MB FLASH is not supported.

10. Configuring TFTP

In order to use PIBS for downloading applications via Ethernet, the host workstation must be configured to support the TFTP daemon. The following section describes steps required to configure the TFTP daemon included with the evaluation kit software.

If you choose to use the TFTP daemon provided with this package, the **tftpaccess.ctl** files on the host must be customized to match system requirements. You may also need to modify or define an ETC system environment variable to specify the location of the **tftpaccess.ctl** file. If the **tftpaccess.ctl** file is located in a directory other than the directory that contains the TFTP program, the ETC environment variable will have to be updated. The ETC environment variable should specify an alternate directory where the **tftpaccess.ctl** file is located.

A sample TFTP daemon control access file, **tftpaccess.ctl**, and the TFTP daemon, **tftpd.exe**, are included in the **\install_dir_path\bin** directory. The **tftpaccess.ctl** file can be modified or copied to another directory and modified appropriately. Entries in **tftpaccess.ctl** describe directories that can be accessed by the TFTP daemon.

When creating or modifying the **tftpaccess.ctl** file, the following rules apply:

- Blank lines and lines beginning with “#” are ignored.
- Each entry must be entered on a single line.
- Each entry must start with the **allow** keyword, followed by:, followed by a directory name. Directory names specify a directory that the TFTP daemon is allowed to access.

For example, to allow the tftpd server access to the **\install_dir_path\platform\sample** directory, the entry below would be suitable.

```
allow:\install_dir_path\platform\sample
```

It is possible to run **tftpd.exe** automatically everytime that Windows is started. To make the program run automatically everytime Windows is started, perform the following steps:

1. Select Start from the Windows task bar.
2. Select Settings.
3. Select Taskbar.
4. Select Start Menu Programs (or Advanced).
5. Select Add...
6. In the command line field enter tftpd.exe. Include the full directory path of the tftpd.exe file if necessary.
7. Select Next.
8. In the Select Program Folder window, select the Programs/Startup folder.
9. Select Next.
10. Select Finished.

The TFTP daemon will start automatically upon the next restart of Windows.

11. IBM Image File Format Description

The IBM image file format is designed to permit the specification of multiple text, symbol and data sections. The format consists of multiple sections prefixed by section headers (of type `rel_block_t`) with a boot header (of type, `boot_block_t`) prefixing the entire file. The boot header specifies the initial target of the image and the image entry point. PIBS software only processes the information in the boot header when loading IBM image files.

11.1 Boot Header

The IBM image file is preceded by the boot header. PIBS loader uses information in the boot header to verify the validity of the IBM image file and to determine the image file destination. The boot header is stripped off by PIBS loader and is not saved in memory. The boot header has the following format:

```
typedef struct boot_block {
    unsigned long magic;
    unsigned long dest;
    unsigned long num_512blocks;
    unsigned long debug_flag;
    unsigned long entry_point;
    unsigned long reserved[3];
} boot_block_t;
```

magic	Identifies this image as a legitimate IBM image and must have the value 0x0052504F.
dest	Target address for the image (after the boot header is stripped off).
num_512blocks	Boot images are padded to a multiple of 512 byte blocks. This field specifies the number of 512 byte blocks.
debug_flag	Currently unused.
entry_point	Specifies the address of the IBM image entry point.

11.2 Section Headers

PIBS places the entire image at the address specified in the boot header. The entry point specified in the boot header is assumed to be a branch, followed by the first section header of type `info_block_t`. This is to allow the bootstrap code to easily gain immediate addressability to the first section header. The format of the first section header is shown below:

```
typedef struct info_block {
    long magic_num;
    long text_start;
    long text_size;
    long data_start;
    long data_size;
    long elf_hdr_size;
    long sym_start;
```

PPC750FX Evaluation Board Kit

```

    long num_syms;
    long toc_ptr;
    struct rel_block *next;
} info_block_t;

```

`magic_num` is used for verification purposes and must be set to 0x004D5054.

`text_start` is the address of the first text section.

`text_size` is the size in bytes of the first text section.

`data_start` is the address of the first data section.

`data_size` is the size in bytes from the of the first data section.

`elf_hdr_size` is currently unused.

`sym_start` is the address of the symbol table.

`num_syms` is the number of symbols.

`next` points to the next section header.

There are three additional section header types that can follow the first section header. Generally, they can occur in the image in any order, but are usually arranged in ascending address order. The section header has the following format:

```

typedef struct rel_block {
    unsigned long type;
    unsigned long dest_addr;
    unsigned long size;
    union {
        struct data_info {
            unsigned long size_to_fill;
            unsigned long char_to_fill;
        } data_info_str;
        struct text_info {
            unsigned long toc_pointer;
            unsigned long entry_pt;
        } text_info_str;
        unsigned long number_symbols;
    } section_info;
    struct rel_block *next;
    struct rel_block *bptr;
} rel_block_t;

```

The **type** field is one of the following manifest constants:

```

#define TEXT_SECT 0x00000001
#define DATA_SECT 0x00000002
#define SYMB_SECT 0x00000004

```

The **dest_addr** specifies the target for the section, while **size** is the extent of the section, not counting the header. The bootstrap program uses this information to move the section to the destination specified at link time. **next** and **bptr** are the section header forward and backward pointers, respectively.

For a text section, the union **section_info** contains the structure **text_info**, specifying the entry point of the text section.

For a data section, the union **section_info** contain the structure **data_info**, specifying **size_to_fill** and **char_to_fill**. These parameters are used to optionally fill a region past the **size** with a character **char_to_fill** for **size_to_fill** bytes.

For symbols, the union **section_info** contains the number of symbols in the section. The data in this section consists of the symbol table from the original object file.





Part III. Sample Applications



12. Sample Applications Overview

This section describes the steps necessary to build, load, and execute the sample application programs included in the evaluation board kit. At this point it is assumed that the serial port and Ethernet board-to-host connections have been made and properly configured. Users can ensure Ethernet connectivity by using the **ping** command from the PIBS shell prompt.

The sample application programs are compiled, assembled, and linked using the GNU for PowerPC or the IBM HighC C++ compilers. EPOS libraries are used during the link step to create an executable file in ELF format. The ELF executable can then be downloaded from the host to the board using the PIBS **bootfile** command.

Note: If you receive an error message regarding the cygwin1.dll file when using the GNU for PowerPC tools, there may be another application on the host that uses a different version of this file. Update your PATH system environment variable (Control Panel->System->Advanced->Environment Variables) so that the GNU tools directory is first in your path.



13. How to Load Application Programs

The PIBS **bootfile** command supports file transfer using a serial port or Ethernet network connection. To load an ELF file using the serial port, issue the **bootfile serial** command from the PIBS prompt, then transfer the ELF file via the kermit protocol using the terminal emulator running on the host. For Windows HyperTerminal users, select Transfer, Send File, set the Filename, Kermit Protocol.

Before loading an ELF file over Ethernet, ensure the board's Ethernet interface has been setup correctly via the PIBS **ifconfig** command. Board to host connectivity can be verified with the PIBS **ping** command or by pinging the board's IP address from the host. After verifying Ethernet connectivity, ensure the PIBS *ipdstaddr0* and *bootfilename* environment variables are set correctly, supplying the host computer's IP address and the directory path/filename of the file to load. To load an ELF file from the host, issue the **bootfile eth** command from the PIBS prompt. PIBS will send a TFTP request for the bootfile to the host using the Ethernet network connection. The host computer must be running a TFTP server/daemon with access to the *bootfilename* specified. See *Configuring TFTP* on page 74 for additional information.

After successful download, PIBS will perform relocation and branch to the program's entry point to start execution. See *bootfile* on page 39 for additional information.



14. EPOS Shell and Dhrystone Sample

To create the EPOS shell sample program, issue the “gnumake ram” command from the **sample** directory. This creates the **finaloram** ELF file suitable for execution on the board. This sample also includes the Dhrystone benchmark which can be run from the EPOS shell. After loading the **finaloram** file using either the PIBS **bootfile serial** or **bootfile eth** command, output similar to that in *Figure 14-1* should be observed.

Figure 14-1. Sample Output from EPOS

```

-----
text start=0x00100000
text end  =0x00136e7c
data start=0x00136e7c
data end  =0x0013d050
bss start =0x0013d050
bss end   =0x0013e8c0
heap start=0x0013e8c0
heap end  =0x001c1740
speed     =400000000
-----
  XXXXX  XX XXX    XXXX  XXXXXXXX
XX   X  XX  XX   XX  XX  XX
XXXXXXXX  XX  XX   XX  XX  XXXXXXXX
XX       XXXXX   XX  XX      XX
  XXXXX  XX      XXXX  XXXXXXXX
        XXXX
-----
Very simple shell for EPOS
type "help()" for help
type "exit()" to quit
EPOS $

```

The Dhrystone benchmark is also included with this sample and can be run by entering **dhry()** at the EPOS shell prompt. Output similar to that below should be observed.

```

Dhrystone Benchmark, Version 2.1 (Language: C)
Program compiled without 'register' attribute
Please give the number of runs through the benchmark:

```

At this point, enter the number of desired iterations. The test is designed not to give results if the selected iterations completes in less two seconds, so pick a large number (≥ 10000000). After the test completes, a check screen will be displayed, followed by the benchmark results. The results may vary based on the system environment, the compiler, and the compiler options used.



15. MultiProcessor Sample

The multiprocessor sample is an EPOS application that can be run on CPU2. PIBS running on CPU1 is used to load the code into the CPU2 application space in SDRAM. The appropriate hardware is then set to execute the code independently on CPU2.

To create this sample program issue the “`gnumake cpu2`” command from the sample directory.

In order to load code images on CPU1 and CPU2, first the CPU2 code image needs to be loaded in SDRAM. This can be done by setting the *bootfilename* variable to point to the **finalocpu2.img** file and by executing the **bootfile ethl2** command. After the **bootfile** command completes execution the CPU2 image will be loaded in memory. After loading the code image file for CPU2 the *bootfilename* variable will have to be changed to point to the **finaloram.img** file. The **finaloram.img** file contains a code image for CPU1. The **finaloram.img** file can then be loaded by executing the **bootfile eth** command. After the **bootfile eth** command completes execution the CPU1 image will be loaded in memory and executed. After the CPU1 image completes initialization and the code running on CPU1 detects that a valid image has been loaded for CPU2 the register in the bridge chip is set allowing the CPU2 to begin execution. Both the CPU1 and CPU2 sample code images execute the EPOS shell. CPU1 image uses serial port 1 and CPU2 image uses serial port 2. CPU2 must use serial port 2 in polled mode due to hardware limitation imposed by interrupt sharing. CPU1 code image uses ethernet 0 and CPU2 image uses ethernet 1. CPU1 and CPU2 code images execute independently. There is no memory coherency with respect to CPU1 and CPU2 images.



16. Debugging Application Programs

PIBS includes a Debug Monitor that allows applications to be loaded and debugged using the RISCWatch debugger in non-JTAG “sim” target mode. The RISCWatch debugger running on the host communicates with the Debug Monitor using the Ethernet connection between the host and the board. The debug connection must be made over Ethernet since debug via the serial port is not supported. The Debug Monitor function is provided by the EPOS debug library (debuglib.a) which PIBS is built with. The Debug Monitor can be included in any EPOS application to provide RISCWatch debug capability.

The debug sample found in the **debug_thread.c** file shows how an application can be debugged using the PIBS Debug Monitor and the RISCWatch debugger in “sim” target mode. The code in the **debug_thread.c** file is linked in the **finalopibs2** image and the **finaloram** image. The code in the **debug_thread.c** file can also be used to create a standalone image file **finalodebug**. It is important to note that when the **debug_thread.c** code is used to create a standalone image file the application code in the **debug_thread.c** file must not use interrupts and must not conflict with PIBS memory space usage. Doing so may cause the PIBS Debug Monitor to RISCWatch debugger connection to fail. If the application program that is being debugged requires the use of interrupts, the EPOS shell sample code (**finaloram** image file) can be loaded on to the board and the RISCWatch debugger can be used to attach to the Debug Monitor in the EPOS shell sample image. Once attached, the Instruction Address Register (IAR) is set to the start of the application code in the **debug_thread()** function. Only application code can be debugged using Debug Monitor. EPOS Operating system code and first level interrupt handlers can not be debugged using Debug Monitor. In order to verify the RISCWatch and Debug Monitor connectivity RISCWatch can also be used to attach to the PIBS Debug Monitor.

Before starting the RISCWatch debugger on the host system, the RISCWatch environment file, **rwppc.env**, must be updated as follows:

```
TARGET_TYPE = sim
TARGET_NAME = x.x.x.x
```

where **x.x.x.x** is the IP address of the evaluation board.

To build the debug sample, issue the “**gnumake debug**” command from the **sample** directory. This creates the **finalodebug** ELF file suitable for execution on the board. After verifying host-to-board Ethernet connectivity by pinging the board's IP address from the host computer, start RISCWatch on the host system. Upon successful connection to the PIBS Debug Monitor, the main RISCWatch command window should appear. If it does not, verify all settings and ensure Ethernet connectivity between the host and the board.

From the RISCWatch command line, use the RISCWatch “**srchpath add pathname**” command to add the **sample** directory to the RISCWatch search path. Issue the RISCWatch “**load file finalodebug**” command to load the file on to the board using the Debug Monitor connection. RISCWatch can now be used as a source level debugger via its connection to the Debug Monitor.

When using RISCWatch with the Debug Monitor code, users should be aware of the following:

- Cache displays are not supported
- Hardware breakpoints are not supported
- Reset command is not supported
- General purpose register R1 should not be manually modified
- Do not use memory that may affect Debug Monitor operation
- Future versions of RISCWatch may use a **TARGET_TYPE** other than “sim” to connect to the Debug Monitor.



Index

A

- about this book, 11
- alias command, 36
- aliaslist, 58
- application tools, 18
- autoboot, 59
- autobootdelay, 60

B

- board connectors, 27
- board reset, 30
- board support software, 18
- boardinfo command, 37
- bootfile command, 39
- bootfilename parameter, 61

C

- cables and power supply, 17
- chipclk command, 42
- chipclcmd command, 62
- commands
 - alias, 36
 - boardinfo, 37
 - bootfile, 39
 - chipclk, 42
 - chipclcmd, 62
 - display, 44
 - echo, 45
 - help, 46
 - ifconfig, 48
 - ping, 49
 - reset, 50
 - route, 51
 - set, 52
 - storefile, 53
 - unalias, 55
 - version, 56
- configuring TFTP, 74
- connecting the board to the host, 27

D

- debug sample, 89
- DHCP
 - switching network connections, 63
- dhcp, 63
- dhcp0/

- dhcp1 parameter, 63
- dhystone benchmark program, 18
- display command, 44

E

- echo command, 45
- embedded PowerPC operating system, 18
- Environment parameters
 - bootfilename, 61
 - dhcp0/
 - dhcp1, 63
 - hwaddr0:hwaddr1, 64
 - ifconfigcmd0:ifconfigcmd1, 65
 - imgaddr, 66
 - ipdstaddr0, 67
 - route cmd, 68
 - userdata, 69
- environment variables
 - aliaslist, 58
 - autoboot, 59
 - autobootdelay, 60
 - dhcp, 63
 - hwdaddr0, 64
 - ifconfigcmd0, 65
 - imgaddr, 66
 - ipdstaddr0, 67
 - route cmd, 68
 - userdata, 69
- epos shell sample, 85
- ethernet setup, 25
- evaluation board kit, 17
- evaluation board platform, 17
- external hardware support, 33

F

- finalopibs2 install file, 73
- finalopibs2.bin install file, 73
- finaloutil install file, 73
- flash_pibs.cmd install file, 73

H, I, J, K

- hardware components, 17
- help command, 46
- host configuration, 25
- host system requirements, 21
 - PC, 21
- how to use this book, 12
- hwaddr0:hwaddr1 parameters, 64
- hwdaddr0, 64
- IBM Embedded Systems Solution Center, 12

PPC750FX Evaluation Board Kit

ifconfig command, 48
ifconfigcmd0, 65
ifconfigcmd0:ifconfigcmd1 parameters, 65
image file format, 75
imgaddr, 66
imgaddr parameter, 66
installation setup, 15
installing the software, 23
ipdstaddr0, 67
ipdstaddr0 parameter, 67

L

loading sample applications, 83

M

multiprocessor sample application, 87

P

PC software installation, 23
 board support package, 23
PC terminal emulation, 29
PIBS, 31
PIBS commands, 35
PIBS install files
 finalopibs2, 73
 finalopibs2.bin, 73
 finaloutil, 73
 flash_pibs.cmd, 73
PIBS memory map, 34
PIBS shell
 user commands
 echo, 45, 62
PIBS software development, 70
 build process, 71
 build tools, 71
 directory structure, 70
 makefile setup, 71
PIBS2 software in FLASH, 72
ping command, 49
PowerPC initialization boot software, 33
PowerPC installation boot software, 18

Q

quickstart, 33

R

Recursion, see Recursion
reset command, 50
RISCWatch debugger, 18
route command, 51
routecmd, 68
routecmd parameter, 68

S

sample applications, 79, 81
 debug, 89
 epos shell, 85
 multiprocessor, 87
serial port
 bootfile parameter, 40, 41
 storefile parameter, 53
serial port setup, 25
set command, 52
software components, 18
storefile command, 53

T

terminal emulator, 29

U, V, W

unalias command, 55
User commands
 echo, 45, 62
userdata, 69
userdata parameter, 69
version command, 56
who should use this book, 11



Revision Log

Revision Date	Contents of Modification
05/06/2003	SA14-2730-00. First draft of the manual.
05/14/2003	SA14-2730-01. Second draft of the manual.
05/15/2003	SA14-2730-02. Third draft of the manual after initial review.
05/19/2003	SA14-2730-03 Fourth version submitted for legal review.
06/15/2003	SA14-2730-04 Added text to support IBM HighC C++ compiler.
09/04/2003	SA14-2730-05 Fixed minor typo in section 11.1. Updated revision log

